Part 3: Design Reviews

References:

- Teorey: Database Modeling & Design, 3rd Edition.
  Morgan Kaufmann, 1999, ISBN 1-55860-500-2, ca. $32.
- Elmasri/Navathe: Fundamentals of Database Systems, 2nd Ed.,
  Appendix A, "Alternative Diagrammatic Notations".
- Rauh/Stickel: Konzeptuelle Datenmodellierung (in German), Teubner, 1997.
- Kemper/Eickler: Datenbanksysteme (in German), Ch. 2, Oldenbourg, 1997.
- Graeme C. Simsion, Graham C. Witt: Data Modeling Essentials, 2nd Edition.
  Coriolis, 2001, ISBN 1-57610-872-4, 459 pages.

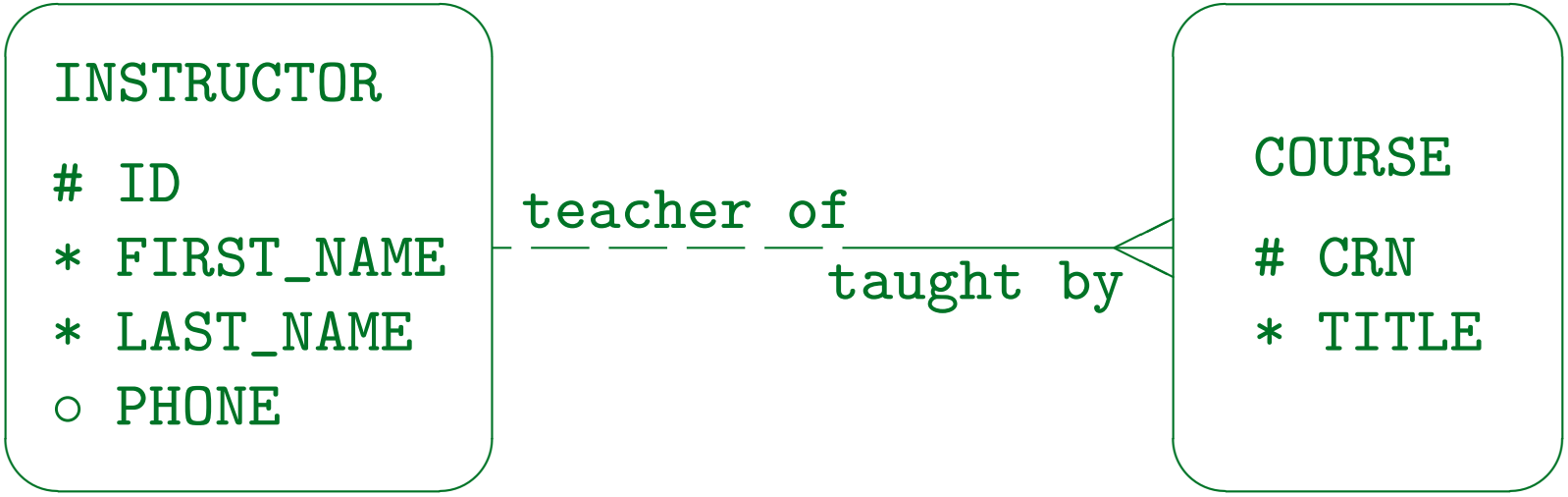
Objectives

After completing this chapter, you should be able to:

- analyze given ER-diagrams for errors.
- check given ER-diagrams for equivalence.
- compare given ER-diagrams, describe advantages and disadvantages, develop questions about the domain of discourse that help to decide between them.

Keys (1)

- Three designers get into a discussion about the right key(s) for instructors.
- Designer A proposes to use an artificial number to identify instructor (attribute ID, "surrogate key"):

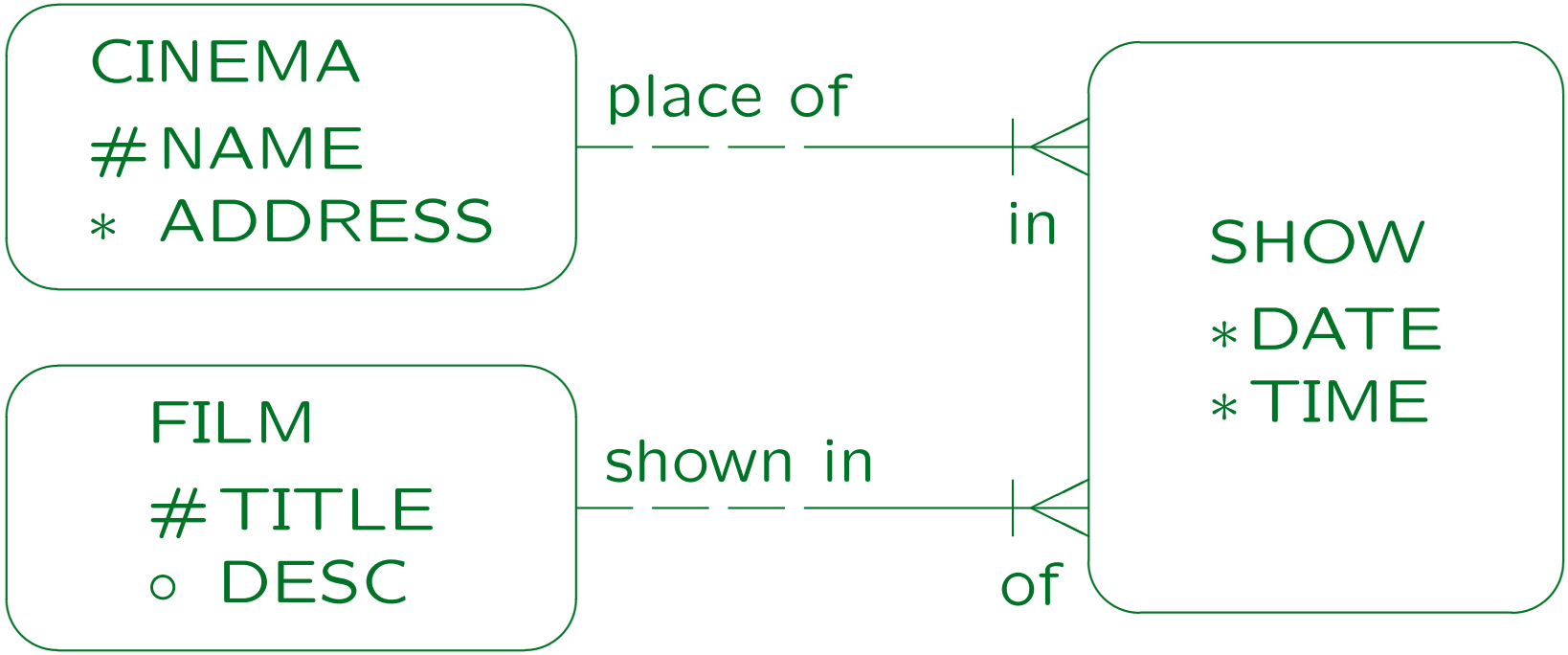


Keys (2)

- Designer B agrees in principle, but proposes to declare in addition "FIRST_NAME" and "LAST_NAME" as alternate key.
- Designer C wants to go even further and remove the artificial ID, and use FIRST_NAME and LAST_NAME together as primary key.
- How would your evaluation of the solutions change if we have to find keys for students, not instructors (e.g. "student is enrolled in program of study")?

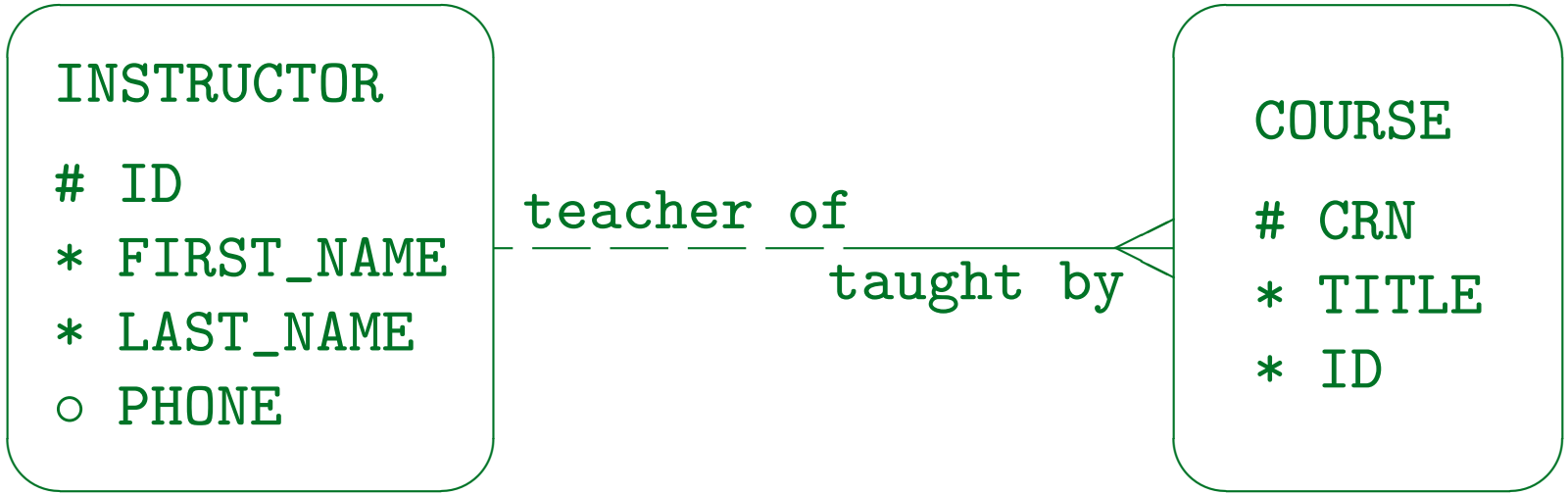
Keys (3)

- Do you see any problem with this model for the cinema program in a city? The same cinema can show the same film several times.



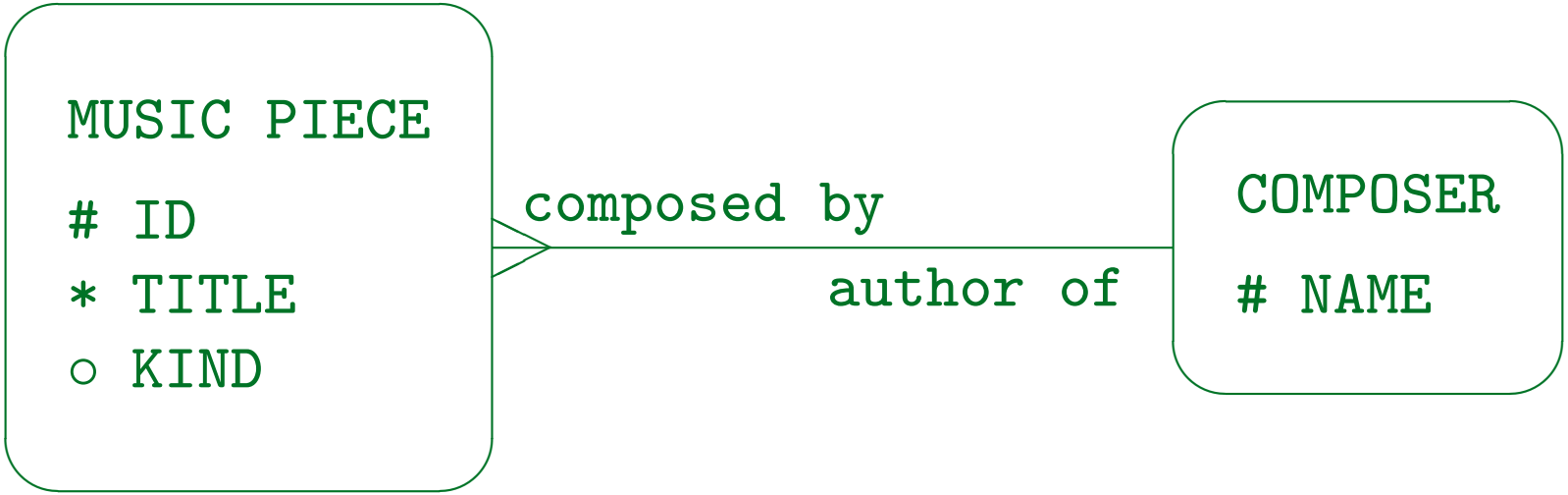
Explicit Foreign Keys

- Coming back to the "instructor teaches course" example, designer D says that it is necessary to put the instructor ID into the course entity type:



Attribute vs. Entity (1)

- In order to store information about music pieces and their composers, designer A comes up with the following schema:

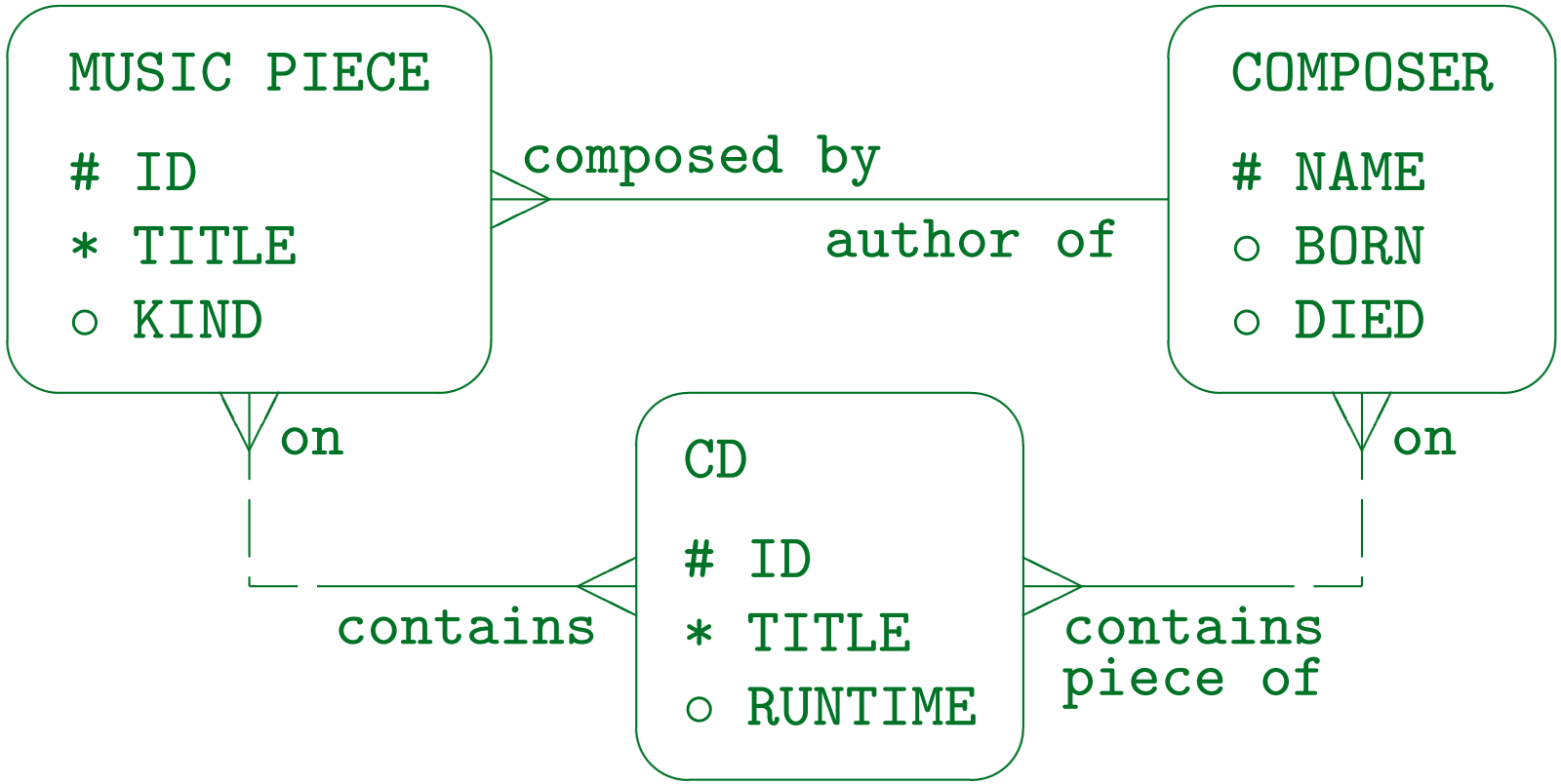


Attribute vs. Entity (2)

- Designer B says that it would be easier to put the composer name as an attribute into the music piece entity type:

Cyclic Relationships

- This schema contains also information about CDs:



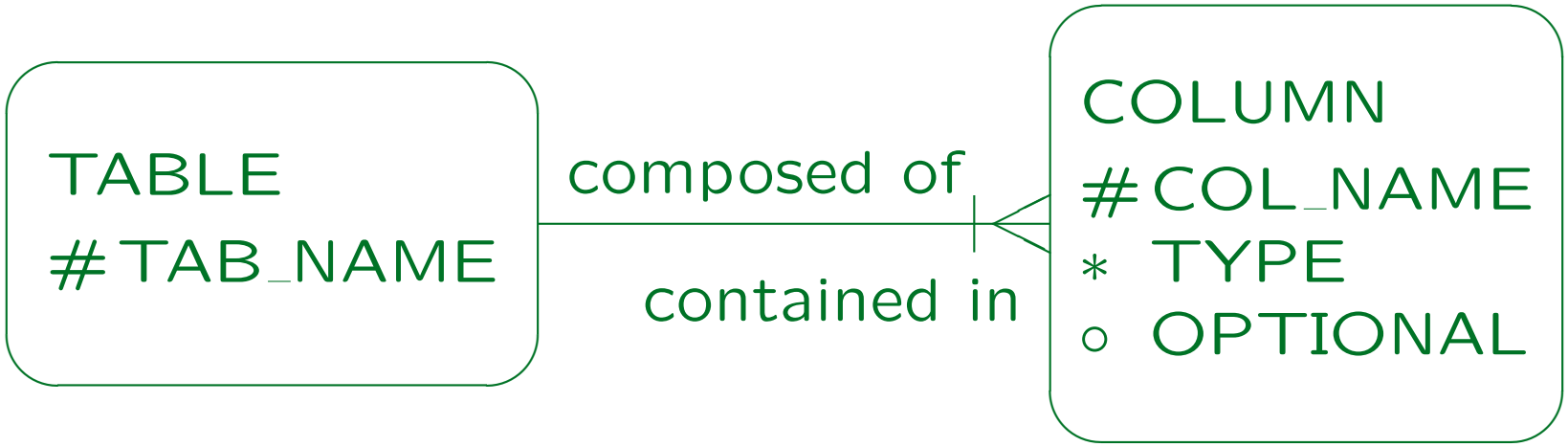
A Small Data Dictionary (1)

- A simple data dictionary of an RDBMS has to be modelled. It must be able to answer at least the following questions:
  - ◇ Which tables exist?
  - ◇ What are the columns of a given table?
  - ◇ What is the data type of a column in a table?
  - ◇ Is a column in a table optional?

    I.e. does it accept null values?
- Different tables can have columns with the same name (with possibly different types/optionality).

A Small Data Dictionary (2)

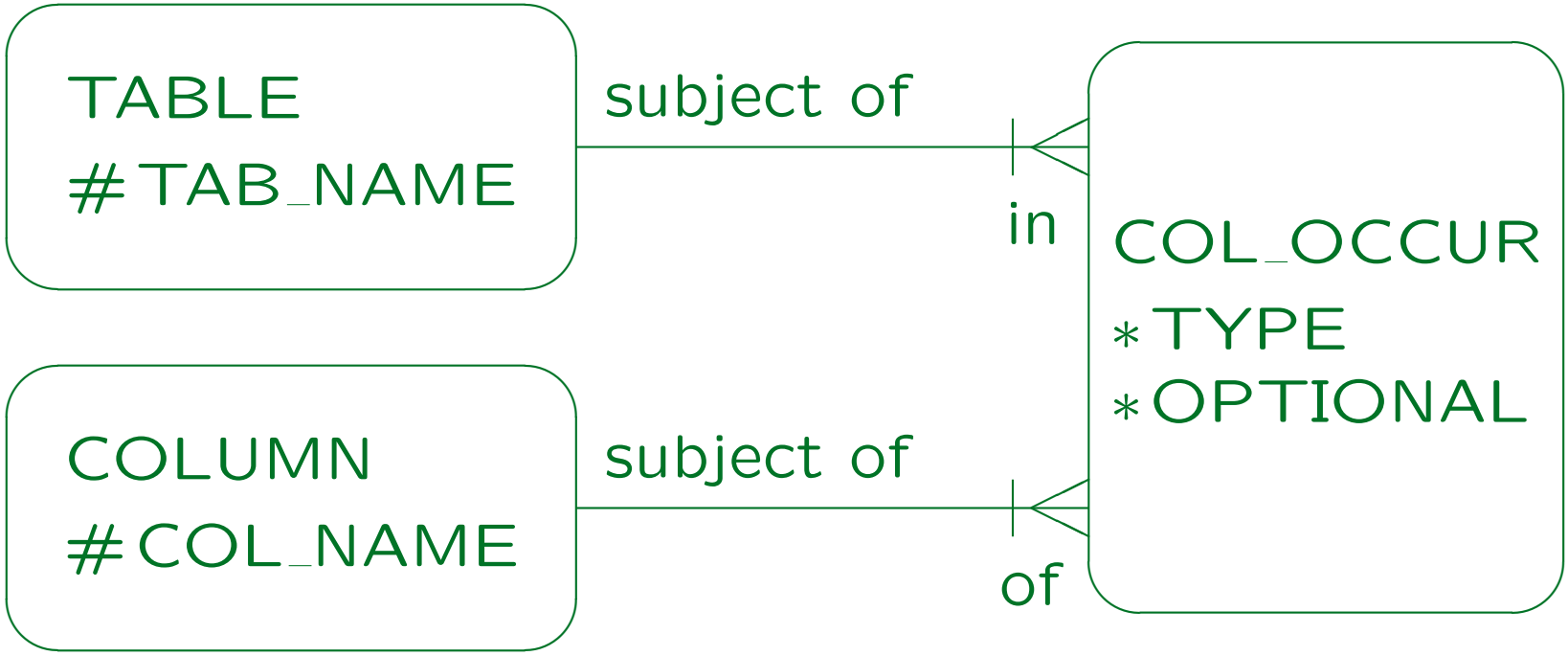
- Designer A proposes to use an entity type for tables and a weak entity type for columns:



- The attribute "OPTIONAL" is constrained to permit only values "Y" and "N".

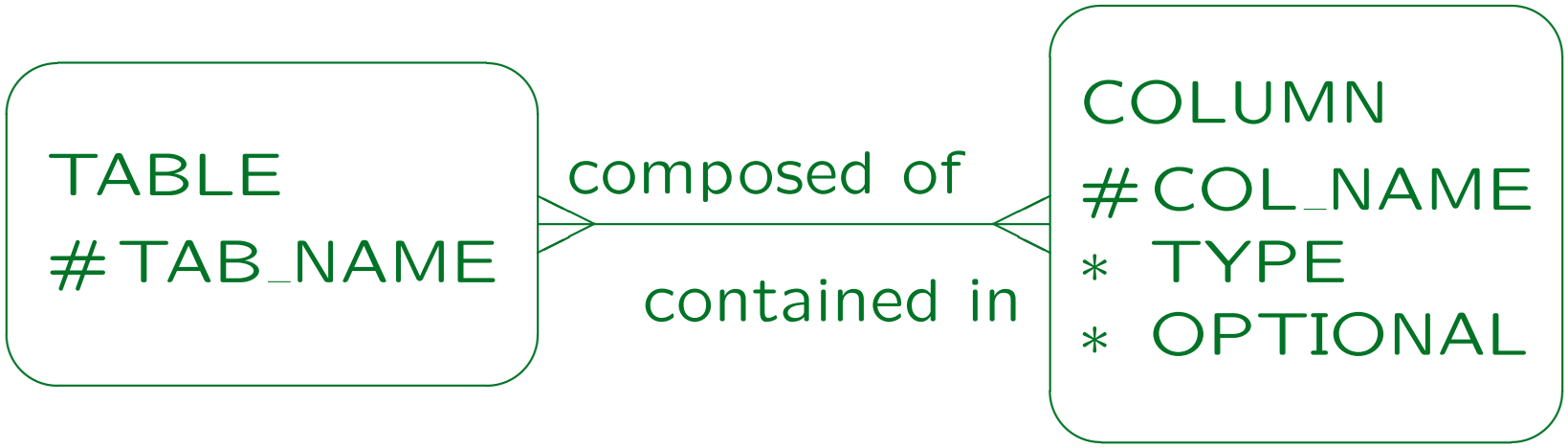
A Small Data Dictionary (3)

- Designer B uses an association entity (he wanted a many-to-many relationship with attributes):



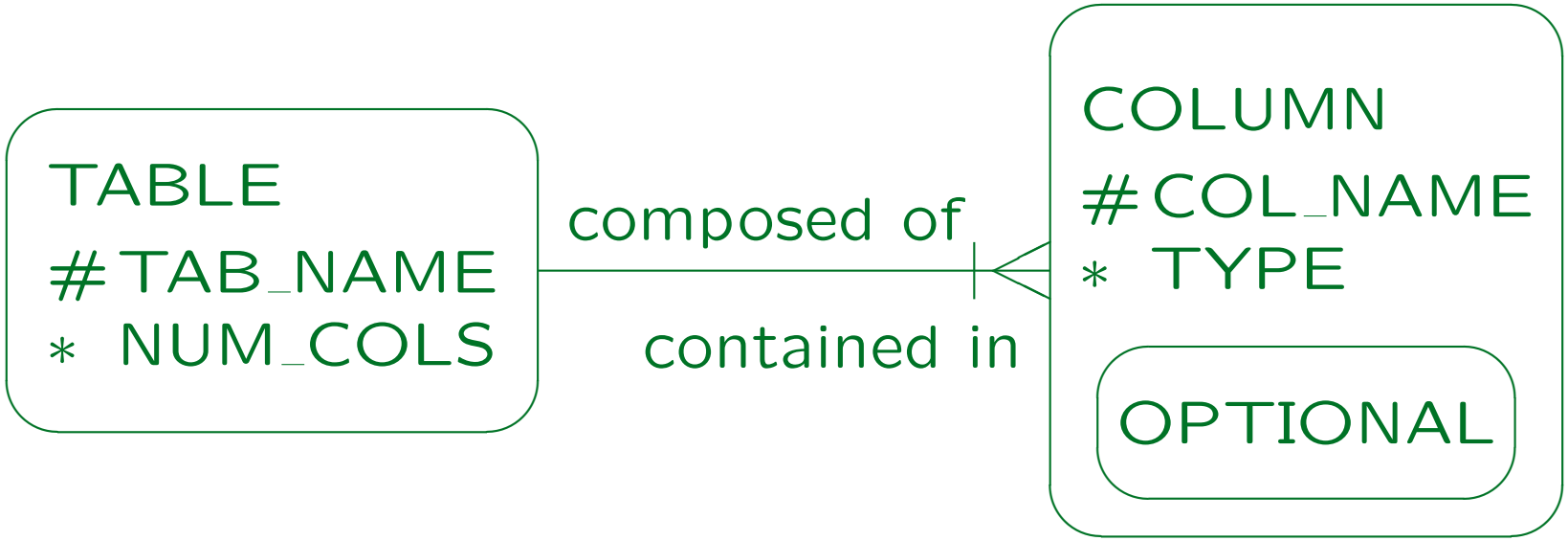
A Small Data Dictionary (4)

- Designer C uses a many-to-many relationship and no weak entity:



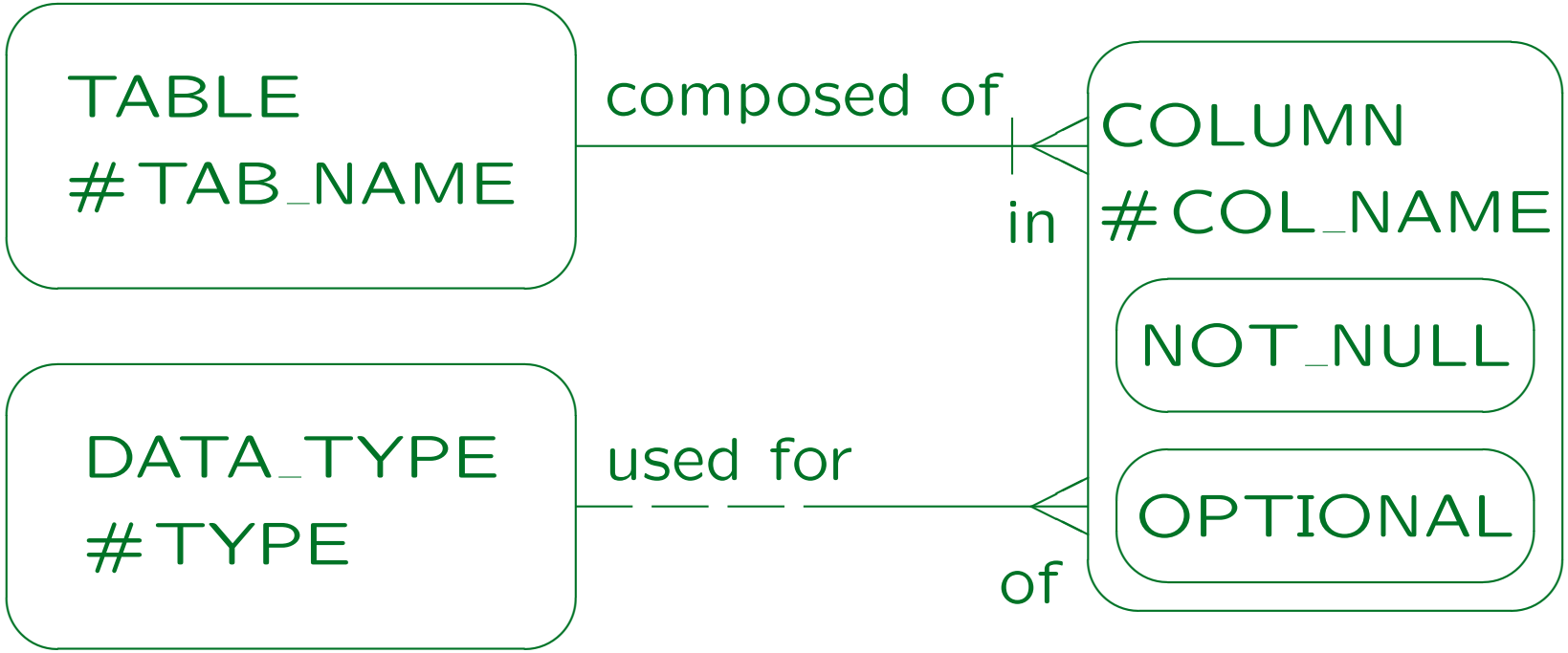
A Small Data Dictionary (5)

- The solution of Designer D is similar to the first one, but he stores the number of columns of a table and also uses a subtype for the optional columns:



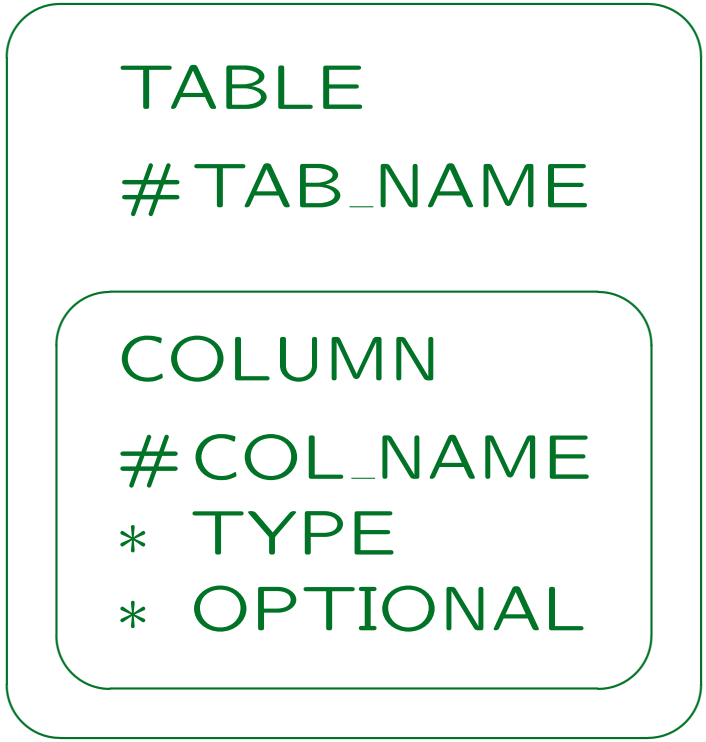
A Small Data Dictionary (6)

- Designer E models the type as an entity:



A Small Data Dictionary (7)

- Designer F models columns as a subtype of tables:

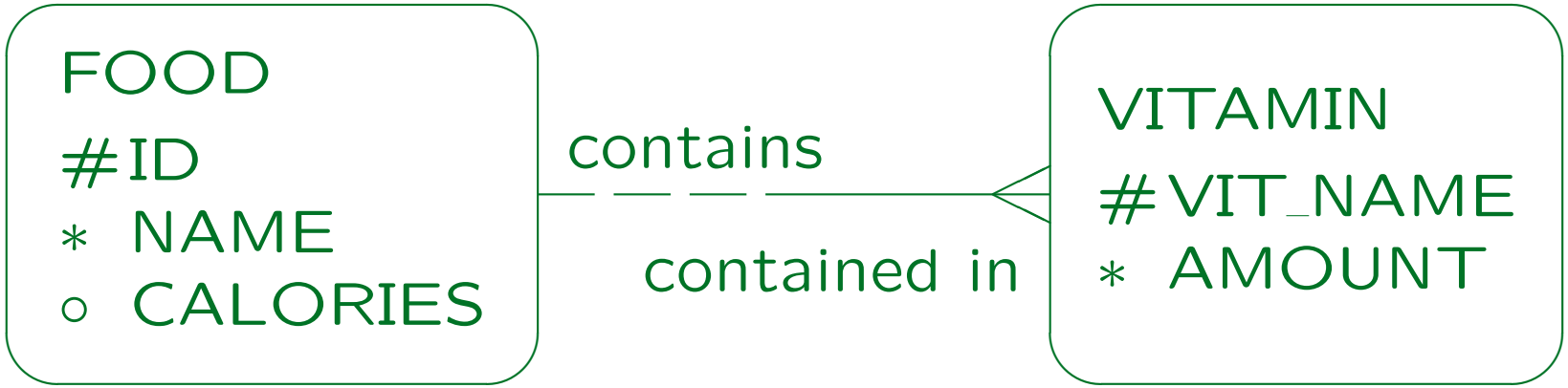


Vitamin Contents (1)

- The task is to store the calories and vitamin contents (per 100g) for various foods.
- Designer A chooses a single entity type:

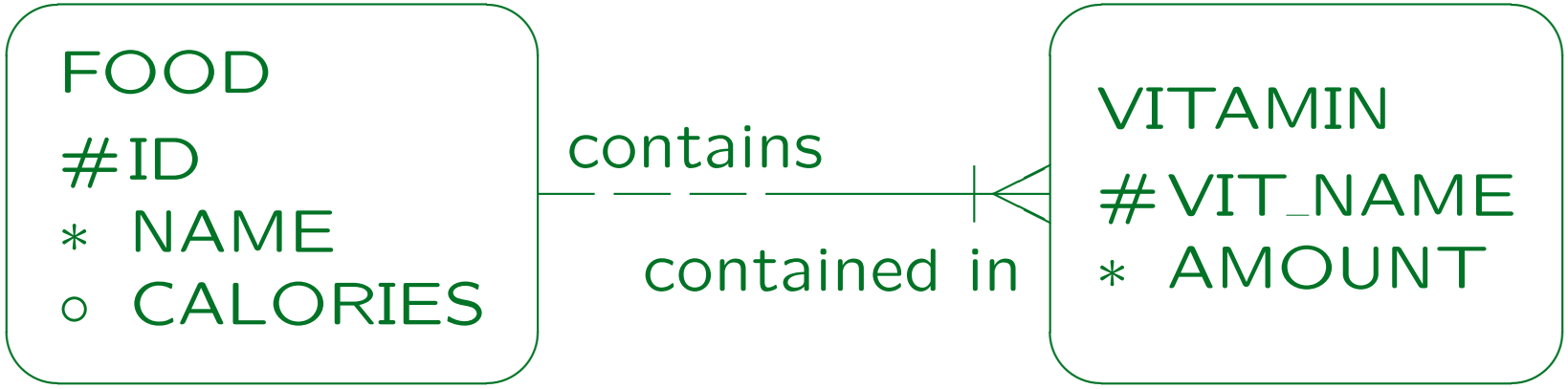
Vitamin Contents (2)

- Designer B uses an entry for each vitamin:



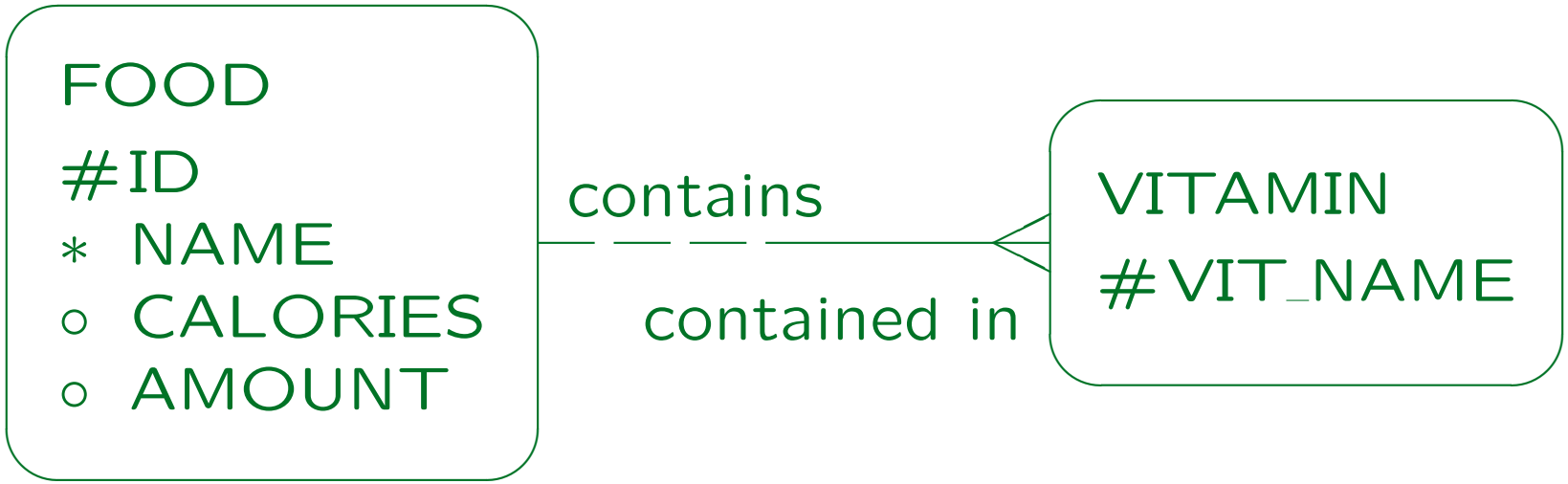
Vitamin Contents (3)

- Designer C uses a very similar solution, however, he declares "VITAMIN" as a weak entity:



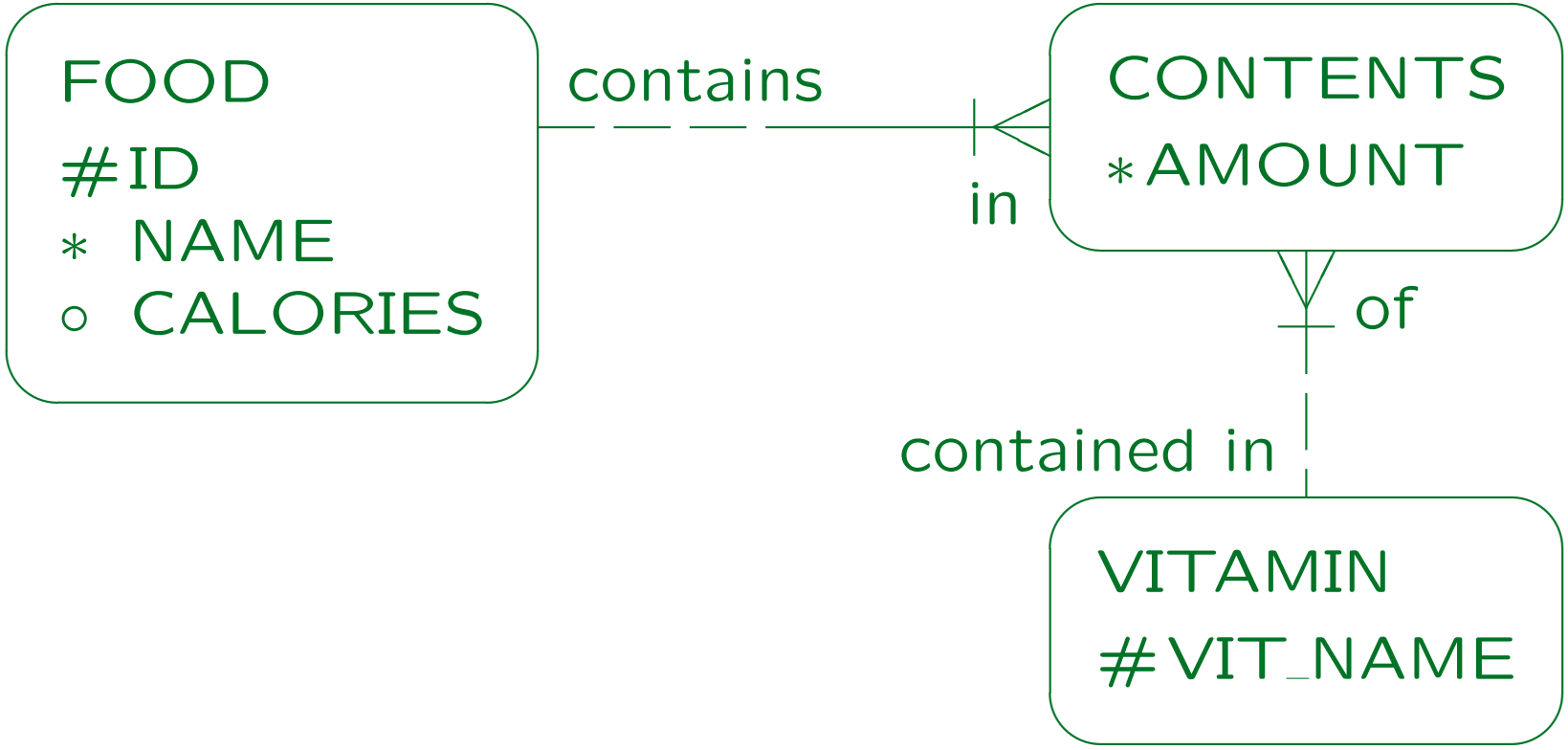
Vitamin Contents (4)

- Designer D uses again a similar solution, but he puts the "Amount" on the "Food" side:



Vitamin Contents (5)

- Designer E proposes a quite complicated model:

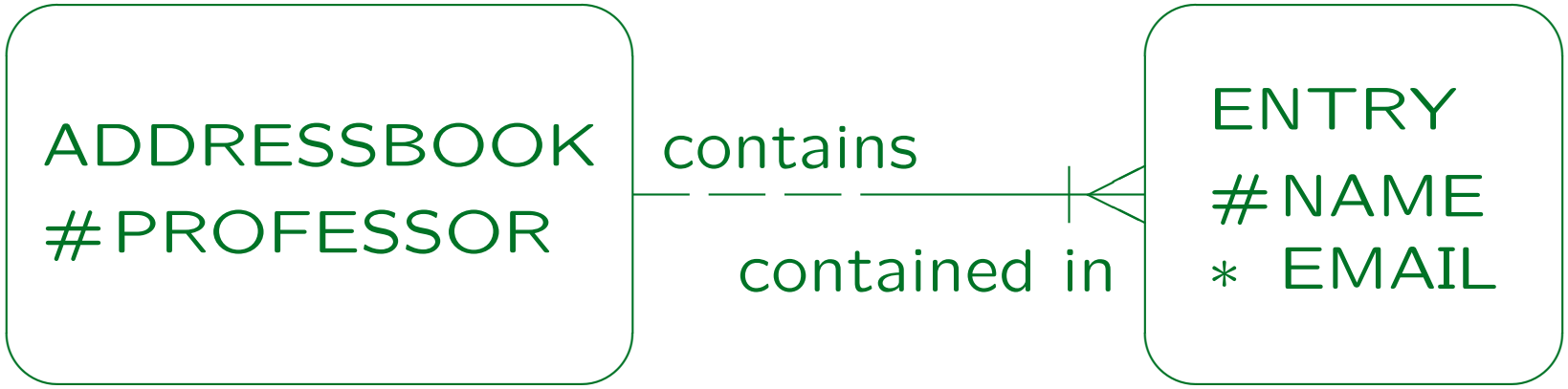


EMail Address Book (1)

- A professor wants to store his EMail address book in a database.
- What are the relative merits of the following solutions?
- Solution A:

EMail Address Book (2)

- Solution B:

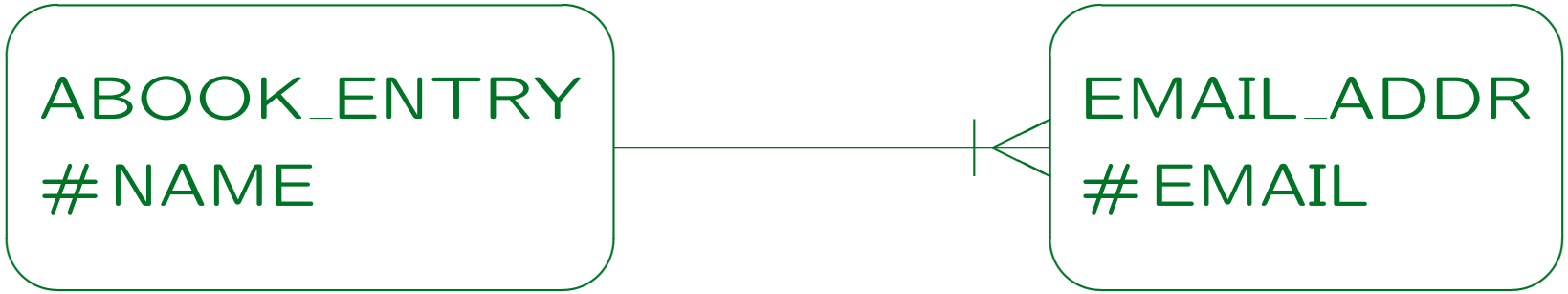


- Solution C:

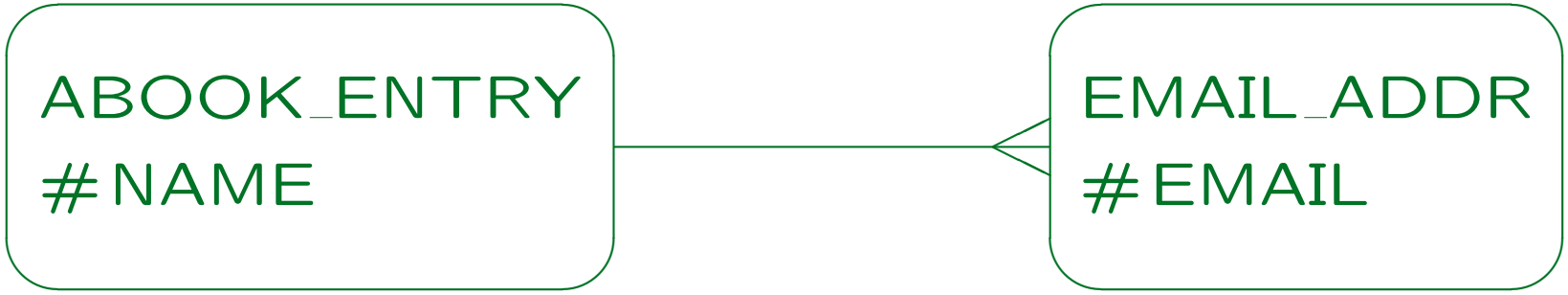
  Like Solution B, but ENTRY is not a weak entity.

EMail Address Book (3)

- Solution D:

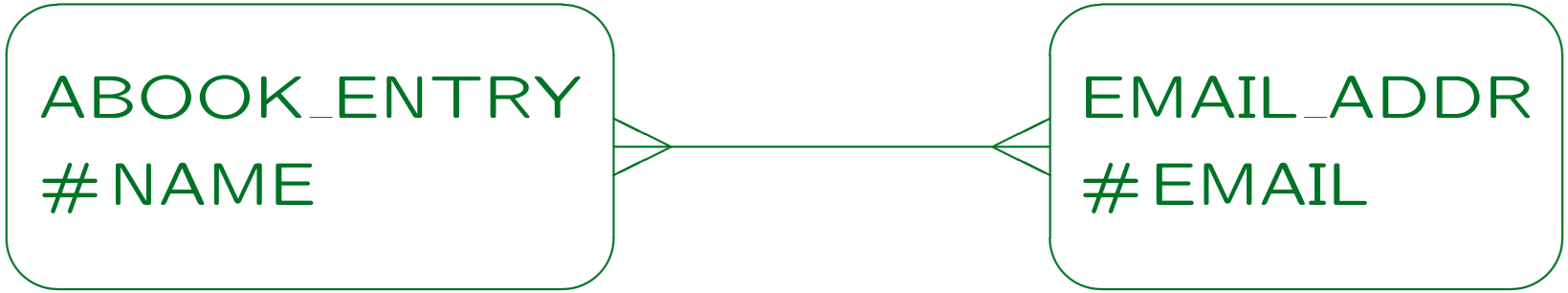


- Solution E:



EMail Address Book (4)

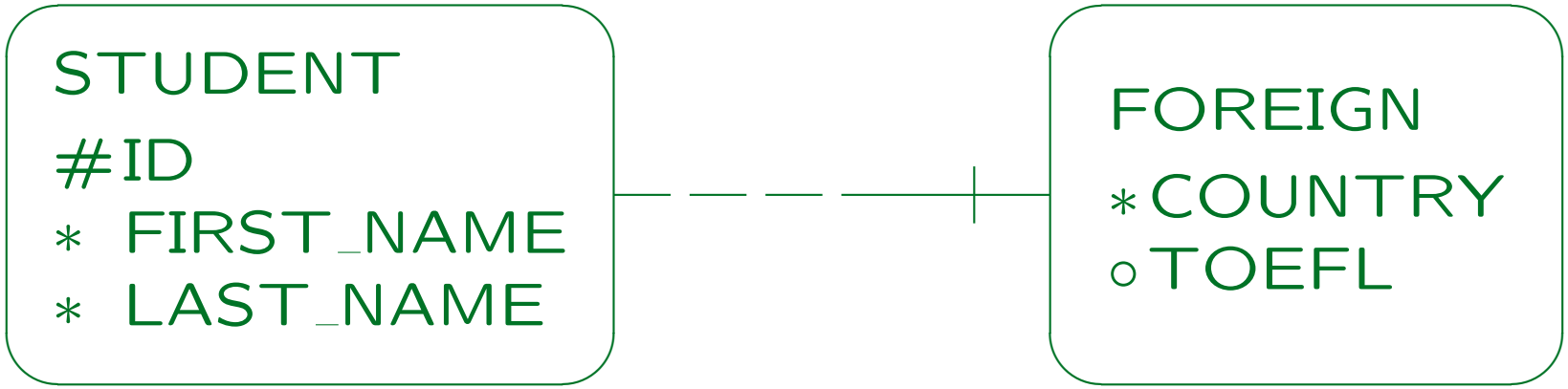
- Solution F:



Student List (1)

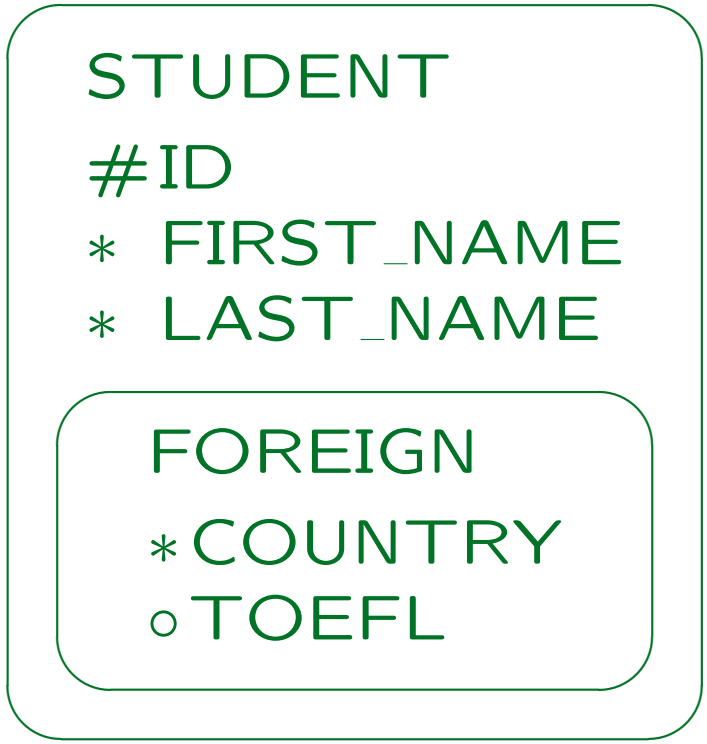
- A university wants to store a list of their students.
- For foreign students, the country and the TOEFL score (Test of English as a Foreign Language) must be kept in addition to the normal data.

  The TOEFL score might not be known when the student data are first entered.



Student List (2)

- Designer B uses a subtype for the foreign students:

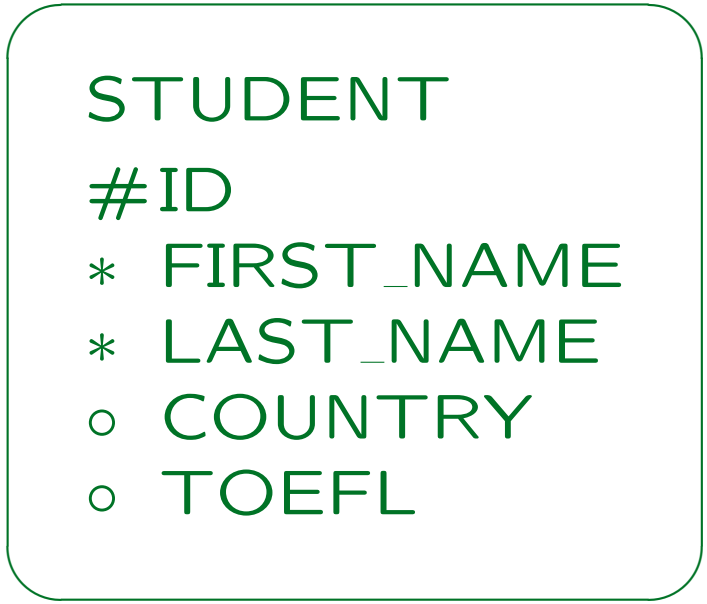


Student List (3)

- Designer C uses two subtypes:

Student List (4)

- Designer D uses a simple entity type:



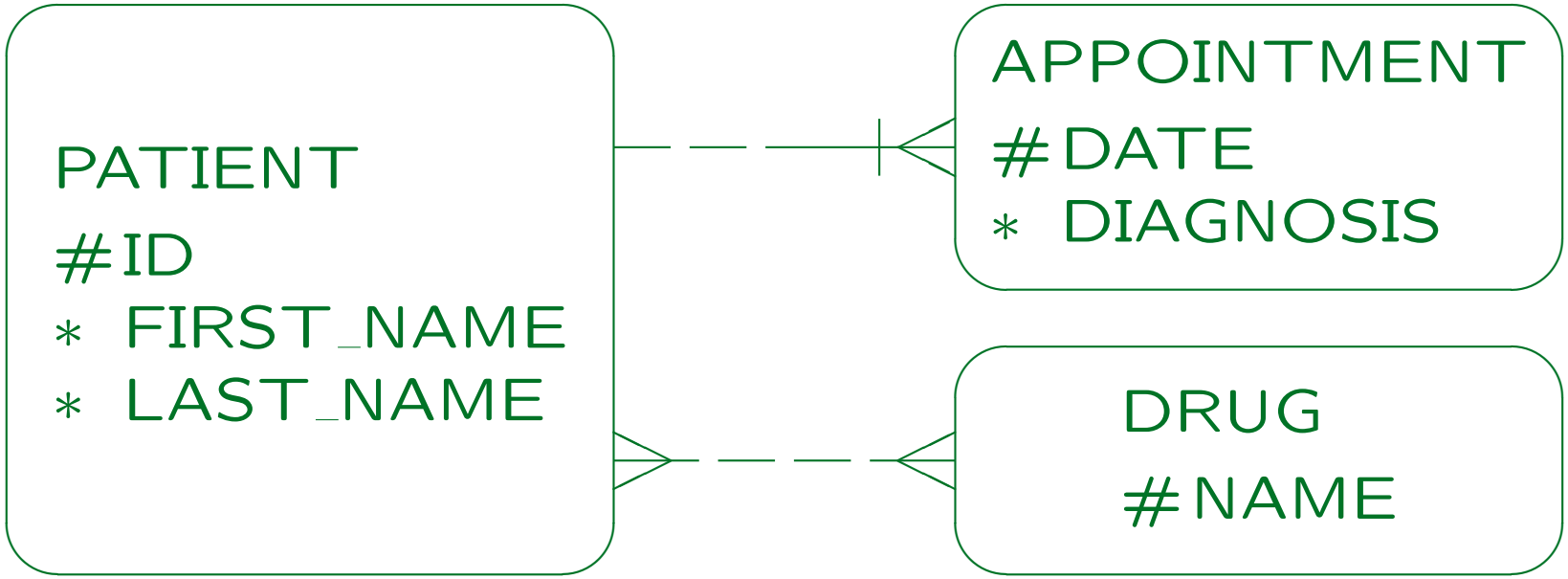
- However, he adds the following constraint:

```
CHECK(TOEFL IS NULL OR COUNTRY IS NOT NULL)
```

Doctor Office

- A doctor wants to keep data about his patients, their appointments, and the prescribed drugs.

  Historical information is important for him: When was which drug prescribed?

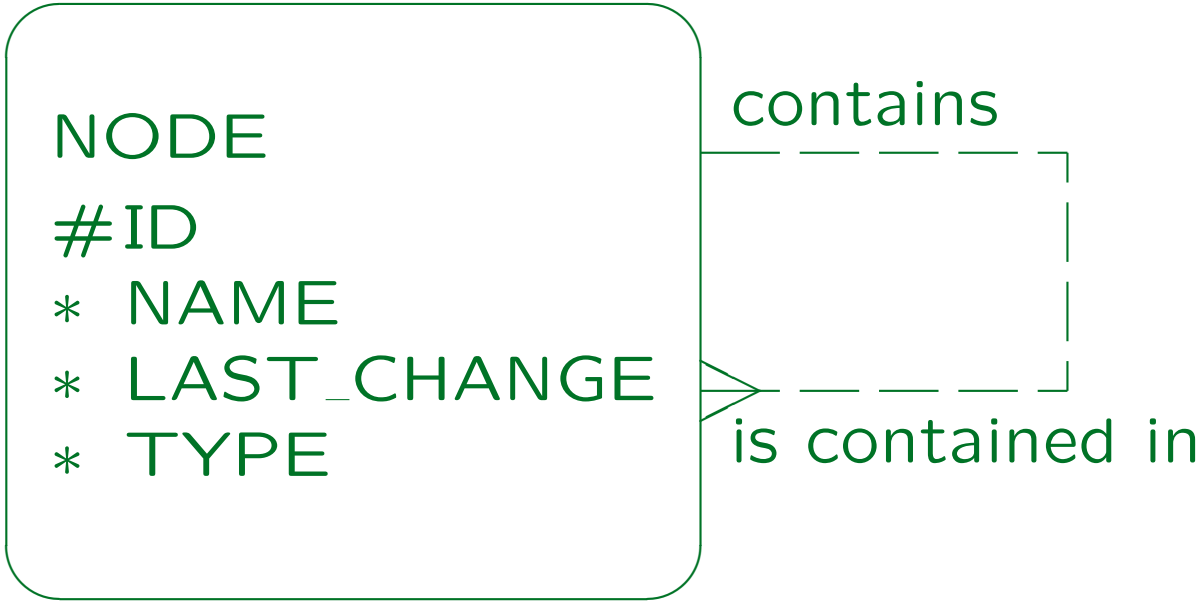


File System (1)

- A file system, i.e. the Unix (or Windows) directory tree should be described by an ER-Diagram:
  - ◇ There are files and directories.
  - ◇ There is a unique ID for files and directories together.
  - ◇ Files have a name that is unique within a directory.
  - ◇ For files and directories, the timestamp of the last change must be stored.
  - ◇ For files, the type must be stored.

File System (2)

Designer A:



- A special value for "TYPE" marks directories.

File System (3)

Designer B: